In the specification:

Please replace paragraph **[0004]** with the following paragraph:

[0004] Interrupts may be raised in hardware by sending a signal down a ~~dedicate~~ dedicated wire, or in software by executing a special instruction. In either case, the processor typically pushes its current register contents onto a stack to preserve them, and starts to execute a new program known as an interrupt handler. When that is complete, the processor may restore its registers from the stack and continue as before. One interrupt may interrupt another, and so on, to many levels.

Please replace paragraph **[0007]** with the following paragraph:

[0007] Multiple levels of interrupts to be utilized in a computer system, which allows, for example, an interrupt with an interrupt level associated with an application to be distinct from an interrupt with an interrupt level associated with a kernel. The kernel level interrupt may be handled quickly via its own handler, while ~~a user level~~ the application level interrupt may be handled more slowly. This may be accomplished by first determining if a first-level handler is installed for the interrupt source. If so, then it may be called. Otherwise, the interrupt source may be masked and a second-level handler may be called. Once this second-level handler has completed its tasks, the interrupt source may then be unmasked. Implementations with three or more levels of interrupt are also possible.

Please replace paragraph **[0021]** with the following paragraph:

[0021] An interrupt source retriever 300 may retrieve the source of the interrupt. An interrupt source active determiner 302 coupled to the interrupt source retriever 300 may then

search the active interrupt sources to determine whether or not they are active. If not, then a method returner 304 coupled to the interrupt source active determiner 302 may return the process from the interrupt. If the source is active, then an installed first level handler determiner 306 coupled to the interrupt source retriever 300 may determine if a kernel-level handler is installed for this source. If so, then that indicates that the interrupt is a kernel-level interrupt, and thus a first-level handler caller 308 coupled to the installed first-level handler determiner 306 may call the ~~kernel-level~~ kernel-level handler. Then the interrupt source may be cleared and the process may await another interrupt.

Please replace paragraph **[0022]** with the following paragraph:

[0022] ~~If it~~ If it was determined that no kernel-level handler was installed for this source, then this may indicate that the interrupt is a user-level interrupt, and thus an interrupt source masker 310 coupled to the installed first-level handler determiner 306 may mask off the interrupt source. An interrupt pending flag setter 312 coupled to the interrupt source masker 310 may set the interrupt pending flag for the source. Then an installed second-level handler determiner 314 coupled to the interrupt source masker 312 may determine if a second-level handler for the interrupt source is installed. If not, then the process may simply continue looking up pending interrupt flags. At this point the interrupt source may be left disabled to prevent the same interrupt source from coming in again. If a user-level handler exists for this source, however, then a second-level handler caller 316 coupled to the installed second-level handler determiner 314 may call the user-level handler. The second-level handler caller 316 may include a pending interrupt flag clearer 318, which may clear the pending interrupt flag. Then an interrupt source unmasker 320 coupled to the second-level handler caller 316 may unmask the interrupt source.

Please replace the Abstract with the following:

ABSTRACT OF THE DISCLOSURE

Multiple levels of interrupts to be utilized in a computer system, which allows, for example, an interrupt with an interrupt level associated with an application to be distinct from an interrupt with an interrupt level associated with a kernel. The kernel level interrupt may be handled quickly via its own handler, while ~~a user level~~ the application level interrupt may be handled more slowly. This may be accomplished by first determining if a first-level handler is installed for the interrupt source. If so, then it may be called. Otherwise, the interrupt source may be masked and a second-level handler may be called. Once this second-level handler has completed its tasks, the interrupt source may then be unmasked. Implementations with three or more levels of interrupt are also possible.

Please update all records accordingly.